Feudal

Relatório Intercalar

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 01:
Jorge Miguel Marques Reis - 080509053
Miao Sun - 080509162

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

7 de Outubro de 2012

Resumo

Neste relatório pretende-se apresentar com detalhe o jogo de tabuleiro "Feudal", e abordar a implementação de uma aplicação para o jogo em linguagem de Programação Prolog.

1 Introdução

Este trabalho tem como objectivo adquirir conhecimentos de implementação segundo o Paradigma da Programação Lógica, desenvolvendo em linguagem de programação Prolog, uma aplicação que permita jogar um jogo tradicional de tabuleiro. Como tal, pretende-se utilizar conceitos lógicos para a recriação de jogadas e implementação da mecânica de jogo segundo as regras de movimentação das peças, e a determinação do fim do jogo com atribuição de um vencedor. O Feudal tem semelhanças ao Xadrez, mas utilizando um mapa com certos terrenos que restringem o movimento. É um jogo para duas pessoas, preparado também para equipas, até 6 jogadores, com regras ligeiramente distintas. No contexto do objectivo deste trabalho, apenas se vai implementar o modo de dois jogadores, mas que permite três tipos de utilização: Humano/Humano, Humano/Computador e Computador/Computador. Assim, é necessário desenvolver métodos para representar o estado do jogo em cada jogada, visualizando o tabuleiro, e permitir a interacção dos jogadores com o mesmo. Será também necessário incluir métodos de inteligência artificial, que permitam desenvolver diversos níveis de jogo para o computador.

O trabalho aqui apresentado está a ser desenvolvido utilizando os interpretadores de Prolog: Swi-Prolog e Sictus Prolog.

As proximas secções deste relatório apresentam:

- A descrição do problema, história e regras do jogo;
- Representação do estado do jogo em Prolog;
- Representação de movimentos de jogo em Prolog;
- Visualização do Tabuleiro em modo de texto com predicados em Prolog;
- Conclusões e Perspectivas para o desenvolvimento do trabalho;
- Código desenvolvido até ao momento.

2 Descrição do Problema

O trabalho baseia-se na implementação do jogo de tabuleiro "Feudal"para computador, utilizando a linguagem de programação Prolog.

O feudal é um jogo estilo Xadrez, originalmente lançado pela 3M Company em 1967. Utilizando recriações de personagens medievais, o objectivo do jogo é invadir e tomar o castelo do adversário ou matar a sua realeza, O Rei, o Príncipe e o Duque, enquanto estrategicamente protege o seu próprio castelo e realeza. O tabuleiro de jogo completo é composto por uma área com 24 por 24 posições (ver Figura 1), divido a meio por um separador no início do jogo

para que cada jogador posicione as suas peças secretamente na sua metade do tabuleiro, respeitando certas regras que serão descritas mais à frente.

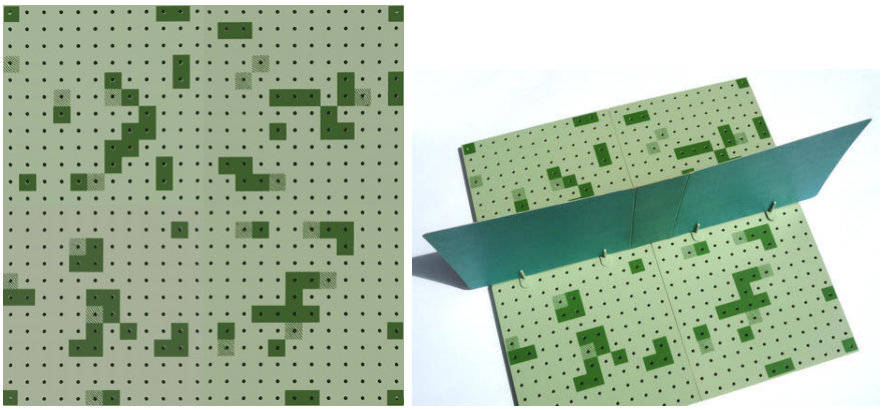

Figura 1: Tabuleiro de Jogo

Cada jogador possui um conjunto de 14 figuras:

- 1 Castle (C)
- 1 King (K)
- 1 Prince (P)
- 1 Duke (D)
- 2 Knights (k)
- 2 Sergeants (S)
- 1 Squire (s)
- 1 Archer (A)
- 4 Pikemen (p)

Ao colocar as peças no tabuleiro, o jogador tem de ter atenção a alguns pormenores. As montanhas e terreno acidentado são areas restritas de jogo. Antes de posicionar os peças, o jogador deve verificar as regras de posicionamento e movimentação para homens montados em cavalos e homens apeados. O castelo pode ser colocado em qualquer posição do seu território, incluindo montanhas e terrenos acidentados. Uma vez posicionado, o castelo não pode ser movido. É necessário também definir uma posição à volta do castelo que consistirá no único ponto de entrada ou saída do mesmo. Homens a cavalo não podem ser colocados nem mover para ou através de montanhas ou terrenos acidentados porém, podem atravessar vales entre os mesmos. Homens apeados também não podem ser colocados nessas zonas especiais, contudo podem atravessar terrenos acidentados e vales entre montanhas. Por último, Archers e Squires são as únicas peças que não podem estar no seu castelo, no entanto podem ser colocados ou movidos sobre a posição de entrada do mesmo.

Pode ser usada a técnica de moeda ao ar para definir que começa primeiro. Durante a sua jogada, cada jogador pode mover quaisquer ou até todas as suas

peças, mas pelo menos uma peça tem que ser movida. Cada peça apenas pode ser movida uma vez por turno, e deverá ser movida consoante as regras de cada peça *(ver Figura 2 - Diagrama de Movimentos)*.

Apenas Squires podem ser movidos sobre espaços ocupados, excepto pelo castelo, e todos os outros homens apenas podem atravessar espaços desocupados. Quando se move para atacar/matar, a peça atacante (excepto Archer) deve mover-se para o espaço ocupado pela peça do inimigo, que sairá do tabuleiro de jogo. Os Archers, em vez disso, podem disparar sobre o primeiro homem numa linha de fogo até três posições de distância em qualquer direcção. No entanto, não podem disparar sobre montanhas ou castelos. Para atravessar a posição de entrada do Castelo, ou para sair ou entrar do castelo, um homem deve parar nessa posição e esperar pelo próximo turno para deixá-la. Note-se que o movimento do Squire torna impossível a este entrar no Castelo inimigo a partir da posição de entrada.

O jogo termina quando o Castelo do adversário é tomado, ou quando toda a sua realeza é assassinada. Para tomar o Castelo, o invasor deve chegar à posição de entrada do Castelo e no próximo passo entrar e ganhar o jogo.

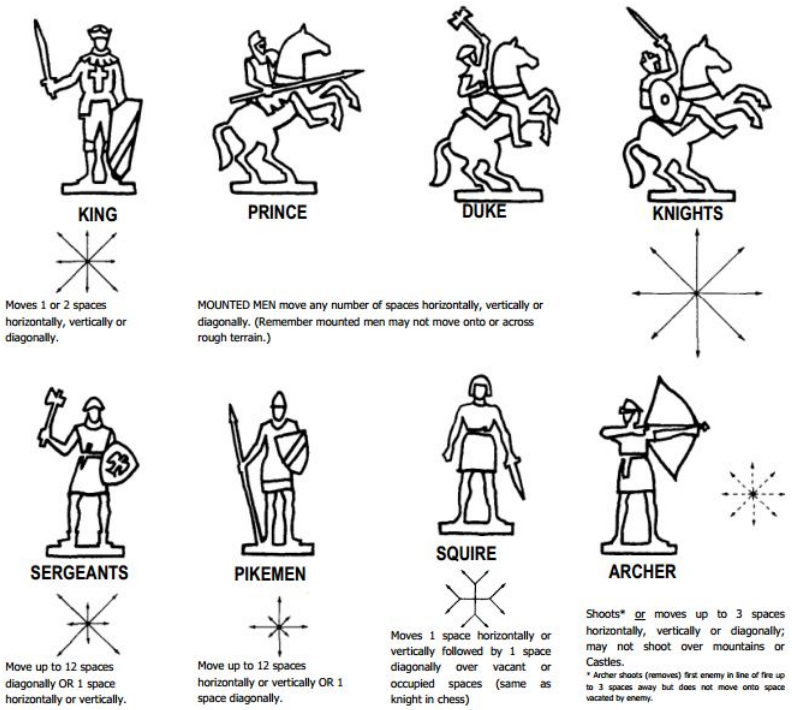

Figura 2: Diagrama de Movimentos

3 Representação do Estado do Jogo

Este jogo desenrola-se num tabuleiro quadrado de 24 por 24 posições, como demonstrado na figura 1, e como tal optou-se por definir o mesmo em Prolog como uma lista de listas.

```
estadoInicial (
[
[ x, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ,
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ t, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, t, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, x, 0, 0, 0, x, 0, 0, 0, t, 0, 0, 0, 0, 0, x, 0, 0, x ],
[ 0, 0, 0, t, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, x, x, x, 0, x, t, 0, x ],
[ 0, 0, 0, x, 0, 0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, x, x, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, x, 0, 0, x, t, 0, 0, 0, x, 0, 0, 0, 0, x, x, t, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, t, x, 0, 0, t, x, x, 0, 0, 0 ],
[ 0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0 ],
[ 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ t, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, t, 0, 0, 0, 0 ],
[ x, x, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, x, x, x, x, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, x, 0, x, 0, 0, x, 0, 0, x, 0, 0, 0, t, 0, 0, 0, 0, 0, x ],
[ 0, 0, 0, 0, x, x, 0, 0, 0, x, x, 0, 0, t, 0, 0, x, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ x, 0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, x ]
]
).
```

A representação anterior corresponde ao estado inicial do tabuleiro, em que "0"representa uma posição vazia, "x"representa montanhas e "t"os terrenos acidentados. Esta é a forma como se encontra o tabuleiro antes do início do jogo, ou seja, antes dos jogadores posicionarem as suas peças, cada um na sua metade vertical do tabuleiro.

A seguinte representação apresenta um possível estado em Prolog do tabuleiro de jogo numa posição mais avançada do jogo, onde já se pode ver posições ocupadas por peças dos dois jogadores, identificadas por letras e números que identificam o jogador e a peça, tal como depois pode ser visto melhor no código disponibilizado no Anexo A.

```
estadoInicial (
[
[ x, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x ],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] ,
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ t, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, t, 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, x, 0, 0, 0, x, 0, 0, 0, t, 0, 0, 0, 0, 0, x, 0, 0, x ],
[ 0, 0, 0, t, 0, a2, t, x, a7, 0, 0, 0, 0, 0, 0, 0, x, x, x, 0, x, t, 0, x ],
[ 0, 0, 0, x, 0, a1, ag, ac, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0 ],
```

[0, 0, 0, 0, 0, a3, 0, x, x, a8, 0, 0, 0, 0, 0, 0, x, 0, 0, x, x, 0, 0, 0],
[0, 0, 0, 0, 0, 0, x, x, 0, a4, a5, 0, a8, a8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, x, 0, 0, a4, a5, 0, a8, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, x, 0, 0, x, t, 0, 0, 0, x, 0, 0, a6, 0, x, x, t, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, t, x, 0, 0, t, x, x, 0, 0, 0],
[0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, b8, b8, b8, b6, 0, x, 0, 0, 0],
[0, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, b5, b4, b8, b7, 0, 0, 0, 0],
[t, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, b4, x, x, t, b1, 0, 0, 0],
[x, x, 0, 0, 0, x, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, b5, x, bg, bc, b2, 0, 0, 0],
[0, 0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, x, x, 0, b3, 0, 0, 0],
[0, 0, 0, 0, 0, x, 0, x, 0, 0, x, 0, 0, x, 0, 0, 0, t, 0, 0, 0, 0, 0, x],
[0, 0, 0, 0, x, x, 0, 0, 0, x, x, 0, 0, t, 0, 0, x, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[x, 0, 0, 0, 0, t, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, x, 0, 0, 0, 0, x]
]
).

4 Representação de um Movimento

Inserção de peças no início do jogo e movimentação das peças durante o jogo são os principais movimentos previstos. Para tal serão implementados predicados que serão responsáveis por essas tarefas.

Predicado para inserir Peça: inserePeca(J, Peca, X, Y, Tab, Tab2) :- ...

'J' representa o jogador que vai inserir, 'Peca' a peça, X e Y as coordenadas da posição a inserir, Tab o tabuleiro prévio e Tab2 o tabuleiro com a peça inserida.

Predicado para mover Peça: movePeca(J, X1, Y1, X2, Y2, Tab, Tab2) :- ...

Neste, X1 e Y1 representam as coordenadas da peça a mover, X2 e Y2 as coordenadas da posição para onde o jogador pretende mover.

5 Visualização do Tabuleiro

Ao ser chamado o predicado "show", é passado 'Tab' como parâmetro ao predicado "estadoInicial", que vai permitir criar o tabuleiro, sendo visualizado com o predicado "print_tab"que recebe essa lista de listas como argumento. De seguida são imprimidos os cabeçalhos e o tabuleiro seguindo o algoritmo apresentado de seguida. Na figura **??** podemos ver o tabuleiro no seu estado mais primitivo, antes do início do jogo, e ainda um exemplo do tabuleiro em modo texto durante o jogo. Podemos claramente ver algumas peças posicionadas no tabuleiro de jogo.

Código para a representação do tabuleiro em Modo Texto:

```
show:-estadoInicial(Tab), print_tab(Tab).
```

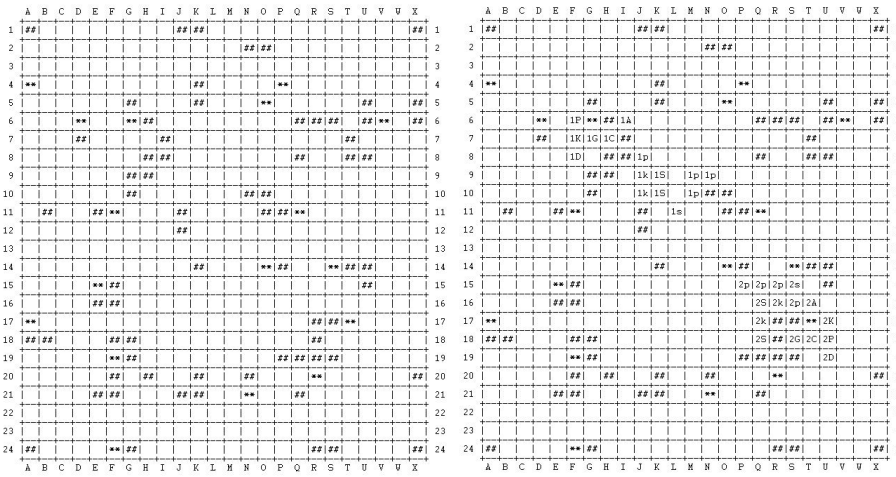

Figura 3: Exemplo de Tabuleiros em Modo Texto no início e durante um jogo

```
% imprime cabeçalhos e chama o predicado para imprimir as listas do tabuleiro
print_tab(Tab):- writeln('    A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X '),
 printLists(Tab,1),
 writeln('    A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X '),nl.

lim:- writeln('   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+').


% imprime todas as listas
printLists([ ],_):-lim, !.
printLists([FList|OList],N):-
                        lim,
        N < 10, write(' '), write(N), write(' |'),
printList(FList),
N2 is N+1, write(' '), writeln(N),
                        printLists(OList,N2),!;

N> 9, write(N), write(' |'),
                        printList(FList),
N2 is N+1, write(' '), writeln(N),
                        printLists(OList,N2).

% imprime todos os elementos de cada sublista
printList([ ]).
printList([FElem|OElem]):-
                        write(''), draw_casa(FElem), write('|'),
                        printList(OElem).

% desenha a casa com os valores respectivos
draw_casa(Elem):-
Elem == 0,  write('  ');
Elem == x,  write('##');
Elem == t,  write('**');
Elem == ac, write('1C');
Elem == ag, write('1G');
Elem == a1, write('1K');
Elem == a2, write('1P');
Elem == a3, write('1D');
Elem == a4, write('1k');
Elem == a5, write('1S');
Elem == a6, write('1s');
Elem == a7, write('1A');
Elem == a8, write('1p');
Elem == bc, write('2C');
Elem == bg, write('2G');
Elem == b1, write('2K');
Elem == b2, write('2P');
```

```
Elem == b3, write('2D');
Elem == b4, write('2k');
Elem == b5, write('2S');
Elem == b6, write('2s');
Elem == b7, write('2A');
Elem == b8, write('2p').
```

6 Conclusões e Perspectivas de Desenvolvimento

Analisando o jogo que se propôs desenvolver e o trabalho já realizado, pode-se concluir que os objectivos iniciais foram atingidos. O jogo "Feudal", devido às suas características de regras de movimentação e condições de terminação baseadas na lógica, pode ser implementado utilizando a linguagem prolog, o que permitirá ser jogado em modo de texto ou mesmo servir de suporte lógico a um ambiente gráfico.

Apesar de ainda em fase muito primitiva, já existe a noção dos algoritmos e predicados a desenvolver que permitirão inserir e movimentar as peças no tabuleiro, realizando as jogadas entre dois humanos ou com o computador.

Visto estar a ser desenvolvido numa linguagem e paradigma que ainda não é familiar aos desenvolvedores e se encontra em aprendizagem, estima-se que ainda falte desenvolver cerca de 85% do trabalho.

Bibliografia:

[1] BoardGameWeek. Feudal Board Game. http://boardgamegeek.com/boardgame/847/feudal.

[2] The Game Pile. Feudal - The Game of Siege and Conquest. Feudal - The Game of Siege and Conquest.

[3] How to Play Feudal - Rules. http://spelarch.khbo.be/PDFspelregels/2922.pdf. Versão digital das regras que acompanham o jogo físico.

[4] Alberto Pacheco. Prolog: Listas (I). http://expo.itch.edu.mx/view.php?f=prog_50#page3.

[5] Bill Wilson. The Prolog Dictionary. 1998-2012. http://www.cse.unsw.edu.au/ billw/prologdict.html.

[6] A. Aaby. Prolog Tutorial. 1996. http://www.lix.polytechnique.fr/ liberti/public/computing/prog/prol tutorial.html.

[7] J. R. Fisher. Prolog :- Tutorial. 2011. http://www.csupomona.edu/ jr-fisher/www/prolog_tutorial.

[8] Loiseleur Michel and Vigier Nicolas. Prolog. 2001. http://boklm.eu/prolog.

A Código Desenvolvido

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%      Feudal em Prolog       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%   Miao Sun  &  Jorge Reis   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%       Turma 3 Grupo 1       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%    PLOG  MIEIC 2012/2013    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:-use_module(library(lists)).

% x - Mountains
% t - rough terrain
% 0 - vazia

%%%%%%%%%% Peças do jogo
%%%Jogador 1
% ac - Castle    x1
% ag - Green     x1
% a1 - King      x1
% a2 - Prince    x1
% a3 - Duke      x1
% a4 - Knights   x2
% a5 - Sergeants x2
% a6 - Squire    x1
% a7 - Archer    x1
% a8 - Pikemen   x4

%%%Jogador 2
% bc - Castle    x1
% bg - Green     x1
% b1 - King      x1
% b2 - Prince    x1
% b3 - Duke      x1
% b4 - Knights   x2
% b5 - Sergeants x2
% b6 - Squire    x1
% b7 - Archer    x1
% b8 - Pikemen   x4


mountedmen([2,3,4,4]).
footmen([1,5,5,6,7,8,8,8,8]).

jogador(j1,1).
jogador(j2,2).


%%%%%%%%  Estado inicial  %%%%%%%%
estadoInicial([[x,0,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,0,0,x],
               [0,0,0,0,0,0,0,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [t,0,0,0,0,0,0,0,0,0,x,0,0,0,0,0,t,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,x,0,0,0,0,x,0,0,0,0,t,0,0,0,0,0,0,x,0,0,x],
               [0,0,0,0,t,0,0,t,x,0,0,0,0,0,0,0,0,0,0,x,x,x,0,x,t,0,x],
               [0,0,0,0,x,0,0,0,0,x,0,0,0,0,0,0,0,0,0,0,0,0,x,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,x,0,0,x,x,0,0,0],
               [0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,x,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,0],
               [0,x,0,0,x,t,0,0,0,0,x,0,0,0,0,0,x,x,t,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,0,x,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,x,0,0,0,t,x,0,0,t,x,x,0,0,0,0],
               [0,0,0,0,t,x,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,x,0,0,0],
               [0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [t,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,x,x,t,0,0,0,0],
               [x,x,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,x,0,0,0,0,0,0,0],
               [0,0,0,0,0,t,x,0,0,0,0,0,0,0,0,x,x,x,x,0,0,0,0,0],
               [0,0,0,0,0,0,x,0,x,0,0,x,0,0,x,0,0,0,t,0,0,0,0,0,x],
               [0,0,0,0,x,x,0,0,0,x,x,0,0,t,0,0,x,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
               [x,0,0,0,0,t,x,0,0,0,0,0,0,0,0,0,x,x,0,0,0,0,x]]).


show:- estadoInicial(Tab), print_tab(Tab).
```

```
show2:-estadoTeste(Tab), print_tab(Tab). % predicado que permite imprimir versões de teste do tabuleiro

print_tab(Tab):- writeln('    A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X '),
                 printLists(Tab,1),
                 writeln('    A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X '),nl.

lim:- writeln('   +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+').


%%imprime a lista inteira
printLists([],_):-lim, !.
printLists([FList|OList],N):-
                        lim,
                        N < 10, write(' '), write(N), write(' |'),
                        printList(FList),
                        N2 is N+1, write(' '), writeln(N),
                        printLists(OList,N2),!;

                        N> 9, write(N), write(' |'),
                        printList(FList),
                        N2 is N+1, write(' '), writeln(N),
                        printLists(OList,N2).


%imprime todos os elementos de cada sublista
printList([]).
printList([FElem|OElem]):-
                        write(''), draw_casa(FElem), write('|'),
                        printList(OElem).

draw_casa(Elem):-  %desenha a casa com os valores respectivmente
        Elem == 0,  write('  ');
        Elem == x,  write('##');
        Elem == t,  write('**');
        Elem == ac, write('1C');
        Elem == ag, write('1G');
        Elem == a1, write('1K');
        Elem == a2, write('1P');
        Elem == a3, write('1D');
        Elem == a4, write('1k');
        Elem == a5, write('1S');
        Elem == a6, write('1s');
        Elem == a7, write('1A');
        Elem == a8, write('1p');
        Elem == bc, write('2C');
        Elem == bg, write('2G');
        Elem == b1, write('2K');
        Elem == b2, write('2P');
        Elem == b3, write('2D');
        Elem == b4, write('2k');
        Elem == b5, write('2S');
        Elem == b6, write('2s');
        Elem == b7, write('2A');
        Elem == b8, write('2p').


start:-
        welcome, show,
        menu_start.


welcome:-
        writeln('**********************************'),
        writeln('*                                *'),
        writeln('*      Bemvindo ao Feudal        *'),
        writeln('*                                *'),
        writeln('**********************************'),nl.

menu_start:-
        writeln('**********************************'),
        writeln('*                                *'),
        writeln('*     Escolhe o mode do jogo:    *'),
        writeln('*                                *'),
        writeln('*       1.Humano VS Humano       *'),
        writeln('*     2.Humano VS Computador     *'),
        writeln('*   3.Computador VS Computador   *'),
        writeln('*                                *'),
```

```
        writeln('*********************************'),nl,
        write('faca a sua escolha: '), faz_opcao(Op),
        %tipo_jogo(Op, J1, J2),
        write(Op), integer(Op),writeln(' um numero'),
        comeca_jogo(Op).

comeca_jogo(Op):-    %para efeito de teste, ainda não está implementado
        Op == 1, writeln('humano contra humano');
        Op == 2, writeln('humano contra computador'), menu_nivel;
        Op == 3, writeln('computador contra computador'), menu_nivel.


/*
tipo_jogo(1,humano,humano).
tipo_jogo(2,humano,computador).
tipo_jogo(3,computador,computador).
*/

opcao_invalida(Op):-
        Op \== 1, Op \== 2, Op \== 3.


faz_opcao(Op):-
        read(Op),
        not(opcao_invalida(Op)),!;
        writeln('opcao invalida'), write('faca a sua escolha: '), faz_opcao(Op).


menu_nivel:-
        writeln('*********************************'),
        writeln('*                               *'),
        writeln('*         Nivel do Jogo         *'),
        writeln('*                               *'),
        writeln('*            1.Easy             *'),
        writeln('*           2.Normal            *'),
        writeln('*            3.Hard             *'),
        writeln('*                               *'),
        writeln('*********************************'),nl,
        write('faca a sua escolha: '), faz_opcao(Op),
        write(Op), integer(Op),writeln(' um numero').


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% estado para testar tabuleiro %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

estadoTeste([[x,0,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,x],
     [0,0,0,0,0,0,0,0,0,0,0,0,0,x,x,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [t,0,0,0,0,0,0,0,0,0,x,0,0,0,0,t,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,x,0,0,0,x,0,0,0,t,0,0,0,0,0,x,0,0,x],
             [0,0,0,t,0,a2,t,x,a7,0,0,0,0,0,0,0,x,x,x,0,x,t,0,x],
             [0,0,0,x,0,a1,ag,ac,x,0,0,0,0,0,0,0,0,0,0,x,0,0,0,0],
             [0,0,0,0,0,a3,0,x,x,a8,0,0,0,0,0,0,x,0,0,x,x,x,0,0,0],
             [0,0,0,0,0,0,x,x,0,a4,a5,0,a8,a8,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,x,0,0,a4,a5,0,a8,x,x,0,0,0,0,0,0,0,0,0],
             [0,x,0,0,x,t,0,0,0,x,0,a6,0,0,x,x,t,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,x,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,x,0,0,0,t,x,0,0,t,x,x,0,0,0],
             [0,0,0,0,t,x,0,0,0,0,0,0,0,0,0,0,b8,b8,b8,b6,0,x,0,0,0],
             [0,0,0,0,x,x,0,0,0,0,0,0,0,0,0,0,b5,b4,b8,b7,0,0,0,0],
             [t,0,0,0,0,0,0,0,0,0,0,0,0,0,0,b4,x,x,t,b1,0,0,0],
             [x,x,0,0,0,x,x,0,0,0,0,0,0,0,0,b5,x,bg,bc,b2,0,0,0],
             [0,0,0,0,0,t,x,0,0,0,0,0,0,0,0,x,x,x,x,0,b3,0,0,0],
             [0,0,0,0,0,x,0,x,0,0,x,0,0,x,0,0,0,t,0,0,0,0,0,x],
             [0,0,0,0,x,x,0,0,0,x,x,0,0,t,0,0,x,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [x,0,0,0,0,t,x,0,0,0,0,0,0,0,0,0,0,x,x,0,0,0,0,x]]).
```